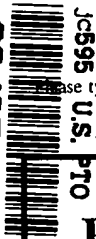


08/27/98



Use type a plus sign inside this box -->



PTO/SB/05 (2/98)
Approved for use through 09/30/00 OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

Attorney Docket No.

Total Pages
(all documents)

First Named Inventor or Application Identifier

Phillip E. Mattison

Express Mail Label
No. EM020278544US

IMPROVING THE PORTABILITY OF
DIGITAL IMAGES

(Only for new nonprovisional applications under 37 CFR 1.53(b))

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO:

Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

1. ☒ *Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)
2. ☒ Specification *Total Pages*
(preferred arrangement set forth below)
- Descriptive Title of the Invention
 - Cross References to Related Applications
 - Statement Regarding Fed sponsored R&D
 - Reference to Microfiche Appendix
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings (if filed)
 - Detailed Description
 - Claims
 - Abstract of the Disclosure
- ☒ Drawing(s) (35 USC 113) *Total Sheets*
☐ Oath of Declaration *Total Pages*
- a. ☒ Newly executed (original copy)
- b. ☐ Copy from prior application (37 CFR 1.63(d))
(for continuation/divisional with Box 17 completed)
- (Note Box 5 below)
- i. ☐ DELETION OF INVENTOR(S)
Signed statement attached deleting
inventor(s) named in prior application,
see 37 CFR 1.63(d)(2) and 1.33 (b).

5. ☐ Microfiche Computer Program (Appendix)
6. ☐ Nucleotide &/or Amino Acid Sequence Submission
(if applicable, all necessary)
- a. ☐ Computer Readable Copy
- b. ☐ Paper Copy (identical to computer copy)
- c. ☐ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. ☒ Assignment Papers (cover sheet & document(s))
8. ☐ 37 CFR 3.73(b) Statement ☒ Power of Attorney
(when there is an assignee)
9. ☐ English Translation Document (if applicable)
10. ☐ Information Disclosure ☐ Copies of IDS
Statement (IDS)/PTO-1449 Citations
11. ☐ Preliminary Amendment
12. ☒ Return Receipt Postcard (MPEP 503)
13. ☐ *Small Entity ☐ Statement filed in prior app
Statement(s) Status still proper and desired
14. ☐ Certified Copy of Priority Document(s)
(if foreign priority is claimed)
15. ☐ Other:

* Note for Items 1 & 13: In order to be entitled to pay small entity fees, a small entity statement is required (37 CFR §1.27), except if one filed in a prior application is relied upon (37 CFR §1.28)

16. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information below & in a preliminary amendment:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application no: _____

Prior application information: Examiner: _____ Group/Art Unit: _____

For Continuation or Divisional Apps only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

NAME: Blakely, Sokoloff, Taylor & Zafman LLP
ADDRESS: 12400 Wilshire Boulevard, 7th Floor
CITY: Los Angeles
COUNTRY: USA

STATE: California
TELEPHONE: (310)207-3800
ZIP: 90025-1026
FAX: (310)820-5988

Name (Print/Type) George G. C. Tseng, Reg. No. 41,355

Signature

Date

August 27, 1998

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL

Note Effective October 1, 1997
Patent fees are subject to annual revision

Complete If Known

TOTAL AMOUNT OF PAYMENT		(\$)\$1,016.00		Application Number	
				Filing Date	8/27/98
				First Named Inventor	Phillip E. Mattison
				Group Art Unit	
				Examiner Name	
				Attorney Docket No.	042390.P4817

METHOD OF PAYMENT (check one)

1. ☒ The Commissioner is hereby authorized to charge indicated fees & credit any overpayments to:

Acct # 02-2666
Acct Name Blakely Sokoloff Taylor & Zafman

- ☒ Charge any add'l fee required under 37 CFR 1.16 & 1.17
☐ Charge issue fee set in 37 CFR 1.18 at the mailing of the Notice of Allowance

2. ☒ Payment Enclosed:

☒ Check ☐ Money Order ☐ Other

Fee Calculation

1. Filing Fee

Large Entity Code	Large Entity Fee (\$)	Small Entity Code	Small Entity Fee (\$)	Fee Description	Fee Paid
101	790	201	395	Utility filing fee	\$790
106	330	206	165	Design filing fee	
107	540	207	270	Plant filing fee	
108	790	208	395	Reissue filing fee	
114	150	214	75	Provisional filing fee	
Subtotal (1)					(\$)

2. Claims

	*	Extra	Fee from Below	Fee Paid
Total Claims	21	20 - 1	X \$22.00	\$22.00
Ind. Claims	5	3 - 2	X \$82.00	\$164.00
Multiple Dependent Claims				

Large Entity Code	Large Entity Fee (\$)	Small Entity Code	Small Entity Fee (\$)	Fee Description
103	22	203	11	Claims in excess of 20
102	82	202	41	Independent claims in excess of 3
104	270	204	135	Multiple dependent claim
109	82	209	41	Reissue independent claims over original patent
110	22	210	11	Reissue claims in excess of 20 & over original patent
Subtotal (2)				

Subtotal (2) (\$)\$186.00

3 Additional Fees

Large Entity Code	Large Entity Fee (\$)	Small Entity Code	Small Entity Fee (\$)	Fee Description	Fee Paid
105	130	205	65	Surcharge-late filing fee or oath	
127	50	227	25	Surcharge-late provisional filing fee or cover sheet	
139	130	139	130	Non-English specification	
147	2520	147	2520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1840*	113	1840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within 1st month	
116	400	216	200	Extension for reply within 2nd month	
117	950	217	475	Extension for reply within 3rd month	
118	1510	218	755	Extension for reply within 4th month	
128	2060	228	1030	Extension for reply within 5th month	
119	310	219	155	Notice of Appeal	
120	310	220	155	Filing a brief in support of appeal	
121	270	221	135	Request for oral hearing	
138	1510	138	1510	Petition to institute a public use proceeding	
140	110	240	55	Petition to Revive-unavoidable	
141	1320	241	660	Petition to Revive-unintentional	
142	1320	242	660	Utility issue fee (or reissue)	
143	450	243	225	Design issue fee	
144	670	244	335	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Petitions related to provisional applications	
126	240	126	240	Submission of IDS	
581	40	581	40	Recording each patent assignment per property	\$40
146	790	246	395	Filing a submission after final rejection	
149	790	249	395	For each add'l invention to be examined	

Other fee (specify)

Other fee (specify)

*Reduced by Basic Filing Fee Paid

Subtotal (3) (\$)\$40.00

SUBMITTED BY

Name George G. C. Tseng, Reg. No. 41,355

Signature 

Date 8/27/98

COMPLETE (if applicable)

Reg. Number

Deposit Acct User ID

*Highest number of claims previously paid for if an amendment is being transmitted.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of

Attorney Docket No. 042390.P4817
Express Mail No. EM020278544US

UNITED STATES PATENT APPLICATION FOR

IMPROVING THE PORTABILITY OF DIGITAL IMAGES

Inventor:

Phillip E. Mattison

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard, Seventh Floor
Los Angeles, California 90025
(310) 207-3800

042390.P4817

BACKGROUND INFORMATION

1. Field of the Invention

This invention is generally related to digital imaging, and, more specifically, to techniques of packaging digital images that may be in different native formats and translating them into a common format.

2. Related Art

The increasing availability of digital imaging devices such as the digital camera has created an unprecedented opportunity for the convergence of the personal computer (PC) industry and the photographic industry. The pictures produced by digital cameras are ideally suited for use with a PC, making the marriage of digital cameras and personal computers seem perfect. There is, however, a significant obstacle to this union.

The current practice in the digital photography business is for manufacturers to develop digital cameras independently of one another. This is not likely to change. Each such new camera is typically bundled with a vertically integrated software package that allows images to be viewed on a host processing system. The software is designed specifically to work with that camera and does not support compatibility with different types of digital cameras or cameras from different manufacturers. There is little or no agreement between manufacturers on how the digital images should be stored aboard the camera, how they should be processed before they are stored, or how the host system (e.g., PC) should handle them.

The closest conventional attempt at a standard for the transfer of images between the device and host is the Twain driver. The Twain driver is a software

module that provides a standard interface to application programs for retrieving digital images from an imaging device. The module translates images from the imaging device's native format into some common format used by the application. Examples of such common formats include the red, green, and blue (RGB) bitmap and the device independent bitmap (DIB) as defined by Microsoft Corp.

Different types of digital cameras often use different native formats for storing digital images. A Twain module specific to a given combination of device, native format, and host operating system (OS) could be stored aboard the digital camera. The Twain module could then be transferred to each new host along with the image data. However, it may not be practical to store such a module aboard a digital camera because of limited memory resources in digital cameras. Also, such modules interact directly with the host operating system, thus creating a security risk as viruses in the camera may be propagated into the host system. Finally, the Twain module is typically written in code that is specific to a particular host processor, making the Twain solution not truly portable.

It would therefore be desirable to have a technique that allows different types of imaging devices that store digital images in different native formats to communicate with a host system to allow the viewing or processing of such images in a common format aboard the host, without having to also load a bulky device-specific Twain driver into the host system.

SUMMARY

The invention in one embodiment is directed at an imaging device having an image sensor for generating sensor data and memory for storing an image object. The image object has image data being related to the sensor data
5 and an image method for being executed by an abstract machine to obtain translated image data based upon the image data.

The above briefly summarized features of an embodiment of the invention, as well as other features and advantages of other embodiments, will be apparent from the following detailed description, claims, and figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is an overall system view of an imaging system according to an embodiment of the invention.

Figure 2 shows steps performed to improve the portability of digital
5 images according to another embodiment of the invention.

Figure 3 shows a host processing system including software and hardware configured according to another embodiment of the invention.

Figure 4 shows part of an imaging device including software and hardware elements configured according to another embodiment of the invention.

DETAILED DESCRIPTION

As briefly summarized above, the invention in one embodiment gives manufacturers of imaging devices such as a digital camera the ability to select their own native format for capturing and storing digital image data, while at the same time making it unnecessary for the host system to have specific knowledge of the particular imaging device that produced the images. The data for an image is stored as part of an object. Thus, each image, rather than being represented as an image file, is packaged as part of an image object in the classical sense of object-oriented software. The image object contains both image data and an associated method. This method is the intelligence needed to support a translation from the native format into a common image file format. Each method defines the translation specific to the native format of its associated image data.

Software in the host system that interprets and executes the image methods may become common to all host processing systems, regardless of hardware and operating system, and can be developed and distributed independent of the imaging devices. Such software is in effect an object execution environment such as a virtual machine. As there is no dependency between the virtual machine and the received image data, new and different imaging devices may be developed independently of developments in the common format and application software. The common format also eliminates the need for application developers to recognize and handle different image file formats currently in use. The image object frees imaging device manufacturers from having to determine what is the best format for storing images aboard the imaging device, or what is the best method for processing them. Such a solution

is particularly advantageous because changes or improvements in the virtual machine and/or the common image file format will probably occur much less frequently than the changes in the imaging devices or the application software.

Figure 1 presents an overall system view of various embodiments of the invention in which image objects are formed or stored in different imaging devices 104a and 104b and storage device 110 before being transferred to a host system 102. Each object 108a, 108b, or 108c includes at least one data portion that is the image data in a format native to the particular type of imaging device that was used to capture the image data. For instance, device 104a may be a scanner configured to generate an object 108a that represents a single image obtained using the device. In contrast, device 104b may be a digital camera configured to generate a single object 108b that has two data portions 114b₁, and 114b₂ representing two still images.

Once the objects are transferred to the host system 102, their methods are interpreted and executed by the abstract machine 120, resulting in data files 116a (from object 108a), 116b₁ and 116b₂ (from object 108b), and 116c (from object 108c). These files will contain digital images in a predefined common format. The data files may then be accessed by application 124 to manipulate or display the image in each data file. The application 124 may be designed to further translate the images into one of numerous other file formats, including Joint Photographic Experts Group format (JPEG), Graphics Interchange Format (GIF), Interchange File Format (IFF), and Tagged Image File Format (TIFF). For convenience, the discussion below will only refer to the interaction between device 104a, object 108a and associated data and methods 114a and 115a. However, it will be recognized that the discussion is not limited to only such an embodiment and

also applies to alternative devices, object, data, and methods, including those shown in **Figure 1**.

The steps performed in one embodiment of the system of **Figure 1** are presented in **Figure 2** to help better appreciate some of the differences between the embodiment of the invention and the conventional technique of using the Twain module to transfer image data from device to host. The technique of creating and transferring portable image data according to one embodiment of the invention may begin with step 204 in **Figure 2** with the capture of raw sensor data. This may be done according to known techniques associated with an imaging device such as a digital camera. Next, image data 114a in a predefined native format is formed based on the raw sensor data. In one embodiment, the raw sensor data is processed by the imaging device 104a according to conventional and/or proprietary operations such as noise removal, color interpolation, image compression, and scaling to obtain data in the desired native format as indicated in the step 208. Alternatively, step 208 may be skipped with no significant processing being performed upon the raw sensor data. In that case, the raw sensor data is considered to be in the native format. The amount of processing in step 208 involves potential performance tradeoffs to be discussed below. Operation then continues with step 212.

In step 212, the imaging device 104a forms an object 108a that includes the image data 114a and an associated image method 115a. The image method 115a is a program or list of instructions to be executed by the abstract machine 120 for translating the image data from the native format to a predefined common format. The method 115a is based on the instruction set of the abstract machine 120. In most cases, the method is expected to be significantly smaller than the

conventional Twain module because the instruction set of the abstract machine can be optimized for image processing and because resource allocation, such as memory allocation, can be built into the abstract machine 120 rather than in the method.

5 After the image object 108a has been formed in the imaging device, operation in **Figure 2** continues with the transfer of the object to the host system. This may be performed using well known communication media for the transfer of image data, such as a wire link, or using a removable storage device 110 such as a non-volatile memory card.

10 The interface between the communication medium and the abstract machine 120 in the host system is the object driver 122. The object driver 122 communicates with appropriate hardware and software (including the operating system) in the host system 102 to make the object 108a available to the abstract machine. The object driver launches the abstract machine under the local OS
15 and eventually causes it to execute the method. In one embodiment, the object driver 122 includes the well known software driver for retrieving files that are configured according to the Tagged Image File Format (TIFF). TIFF is a popular format for storing bit-mapped images on PC's and Macintosh^{AE} computers and can be used to implement the object 108a. Other techniques for implementing
20 the object 108a and its associated driver 122 may be used as recognized by one of ordinary skill in the art.

A loader (not shown) loads the method into the abstract machine and passes it a pointer to the image data. The loader also recognizes certain fields of the image object such as the image count and iteratively initializes the abstract
25 machine to process each image in turn as in step 220. This results in translation

and processing of the image data 114a into translated image data in a common format. The translated data is normally placed in a separate data file 116a and may be stored in a mass storage device such as a hard disk drive. This data file 116a may then be further processed into a desired finished form by the
5 application 124.

Having described an embodiment of the invention from an overall system point of view, **Figures 3 and 4** illustrate other embodiments of the invention as imaging device 104a and the host system 102. **Figure 3** is an embodiment of the host processing system 102 including software and hardware
10 configured to accept and process the image object 108a. This particular embodiment is centered around a bus 332 to which a processor 334 is coupled. The system 102 may be a conventional personal computer having an Intel[®] processor and a Microsoft[®] Windows[®] graphical user interface and operating system. Although the operating system software 340, the object driver 122, and
15 the abstract machine 120 are shown as being separately loaded in memory 328, one of ordinary skill in the art will recognize that these software components may at times be combined in whole or in part, and distributed between the memory 328, the mass storage device 338 (e.g., magnetic rotating disk), and a portable memory such as an optical compact disc (not shown).

20 The processor 334 executes instructions stored in memory 328 (e.g., random access memory (RAM)) and mass storage device 338. The memory 328 is any machine-readable medium such as a semiconductor integrated circuit that may be loaded with instructions that when executed by the processor 334 cause the steps of configuring the host processing system 102 to receive and process
25 objects from different imaging devices and storage devices.

The host processing system 120 also contains a communication interface 316 that complies with any one of conventional computer peripheral bus standards, including RS-232 serial, universal serial bus (USB), and IEEE Standard 1394-1995. In addition to communication via a physical wire, wireless communication media are also contemplated using, for instance, infrared or radio frequency for data transmission. As another alternative to the communication medium, a number of different removable storage devices are contemplated, including Personal Computer Memory Card International Association (PCMCIA), Flash Miniature Card by Intel[®] and any other non-volatile storage media suitable for transporting digital image data.

The host processing system 102 of **Figure 3** also features a user display device interface 342 such as a graphics adapter card that interfaces any conventional display device including a cathode ray tube (CRT), liquid crystal display (LCD), and any other display technique suitable for viewing electronic images. Of course, the host processing system 102 may also feature additional interfaces (not shown) for performing additional I/O (e.g., network interface controller).

Having discussed the host portion of a system embodiment of the invention, **Figure 4** illustrates the imaging device 104a including its software and hardware elements. Once again, the embodiment of **Figure 4** is based on a bus 432 to which a processor 424 and memory 428 are coupled. An alternative here is a microcontroller that supplants the processor 424 and memory 428. The memory may be a machine readable medium such as semiconductor integrated circuit RAM or a non-volatile semiconductor memory such as read only memory (ROM). The memory 428 is loaded with instructions to be executed by

the processor 424 that cause the combining of image data 114a and method 115a as part of an image object 108 (see **Figure 1**). Such instructions are collectively indicated as object formation software 412 in **Figure 4**. The image method 115a is normally developed as a separate piece of software as described below.

5 The memory 428 may also include optional signal processing software 408 that is used to process raw sensor data received from the image sensor 404 into the image data 114a in a native format. Image processing logic circuitry 436 may also be included in the imaging device 104 for obtaining better performance when processing the raw sensor data. The logic circuitry 436 may be integrated
10 into the die of the image sensor 404, which in one embodiment may be a complimentary metal oxide semiconductor (CMOS) active pixel sensor.

 Although not shown in **Figure 4**, the memory 428 also stores software to be executed by the processor 424 for accessing the interfaces 416 and 420 to communicate with devices outside the imaging device 104a. Interface 416 to a
15 communication medium may be one that complies with a computer peripheral bus standard that was discussed above in connection with the host system. Similarly, the interface 420 may connect a removable storage device such as a flash memory card for non-volatile storage and transportation of the image object 108a.

20 The embodiment of the invention in **Figure 4** uses a primarily software-based processor and bus architecture for forming and transferring the image object 108a to the host system 102. An alternative to this could be a primarily hardware-based architecture based on gate arrays and application specific integrated circuits (ASICs) which perform the same functions described earlier in
25 forming and transferring the image object 108a.

The Abstract Machine and the Image Object

The remaining portion of this disclosure discusses techniques for implementing the image object 108a and the abstract machine 120, including a set of instructions that may be used in constructing the image methods 115a.

5 As discussed earlier, the object 108a may be implemented as a Tagged Image File Format (TIFF)/Electronic Photography (EP) structure or file. The TIFF file includes a file header that points to the location of the image data 114a and associated method 115a. The TIFF file and its file header can be accessed by the object driver 122 and made available to the abstract machine 120.

10 The procedure for developing the image method 115a may begin by the developer of the imaging device 104a comparing the predefined native and common formats. An algorithm is developed to translate the image data 114a from the native format to the common format. Such algorithms are either well known or can be easily developed by those of ordinary skill in the art. The
15 algorithm is then implemented in a program using a high level programming language such as C and/or a low level instruction set of the abstract machine 120 described below. The program is then tested and optionally optimized. The program is then compiled into byte code that represents the low level instructions. This becomes the desired image method 115a. The method is then
20 stored in non-volatile memory (e.g., memory 428) aboard the imaging device 104a as part of the object formation software 412 (see Figure 4).

The storage overhead required by the image object 108a residing in the imaging device 104a may be of concern in portable applications such as digital cameras where storage area is typically at a premium. The overhead depends on,

(1) the amount or complexity of processing required to transform the image data 114a from its native format to the common format, and (2) the size of the data 114a in its native format. By selecting a native format that is computationally close to the predefined common format, the image method 115a becomes a relatively short list of simple mathematical or data movement instructions. In comparison, the method 115a is likely to be relatively large if needed to decompress the image data 114a from a highly compressed native format (e.g., JPEG or fractal technology) into the common format. In the latter case, any gains in storage space obtained by compressing the image data are mitigated by the more complex and lengthy method needed for decompressing the image data.

To address the concern for storage space, the abstract machine's instruction set based upon which each method is formed may be highly optimized. For instance, the locations of input buffers (for receiving the image data 114a) and output buffers (for storing the translated data in the common format) that may be used by the abstract machine 120 can be implicit. Address calculations and the majority of the data movement may be performed implicitly by the abstract machine 120 rather than by specific instructions in the image method 115a. The method thus need only concentrate on the algorithm needed to translate its associated image data 114a into the common format.

In addition to optimizing the instruction set to keep the image method 115a compact, another way to save storage area aboard the imaging device is to associate a single copy of the method 115a with multiple sets of image data, as shown in object 108b of **Figure 1**. In this way, the single image object 108b contains multiple images in the image data 114b₁ and 114b₂, becoming in effect

like a conventional roll of film where executing the method 115b would be roughly equivalent to developing the film.

In addition to storage space, another concern for the imaging device manufacturer may be protecting the intellectual property in image processing methodologies. Such processing may be performed on raw sensor data to improve final image quality, e.g. interpolating color filter array patterns, gamma correction, white balance algorithms. Rather than fill the method 115a with these image processing algorithms, thus exposing the algorithms to potential competitors when the object 108a is transferred to the host system 102, the manufacturer of the imaging device 104a may wish to have key image processing functions performed inside the imaging device. This results in image data 114a that are strongly processed versions of the raw sensor data. Taking this to one extreme, those manufacturers who wish to strongly protect their image processing technology could embed all of such processing within the imaging device, resulting in their image data 114a being in the common format. In that case, the corresponding method 115a does nothing but copy the data 114a to an output buffer in the host system 102. At the other extreme, image processing inside the imaging device 104a may be kept to an absolute minimum by simply not performing any processing on the raw sensor data before packaging the image data 114a into the object 108a. The method 115a will define all processing needed to reach the common format, and the abstract machine 120 in the host system performs all of such processing to achieve image data in the common format.

Virtual Machine Architecture

One example of an abstract machine 120 is a virtual machine. The virtual machine is a self-contained operating environment that behaves as if it were a separate computer that executes the image method 115a. The virtual machine helps shield the host system from any viruses that may have been in the received object 108a. The virtual machine may have an instruction set, a register set, and a model of memory independent of the hardware platform of the host processing system 102. An example of an available virtual machine with a well-known interpreter, programming languages, and related tools is the JAVA™ virtual machine and programming language.

In general, the virtual machine should be optimized to allow relatively small code to be written for the image method 115a. Other desirable features are a simple virtual machine interface and speedy code compilation, using, for instance, the known technique of just-in-time code translation, for interpreting the method 115a. The virtual machine should allow dynamic memory allocation to create buffers for temporary data storage while executing the method 115a. The overhead required in the method 115a for memory management can thus be minimal. Also, the host system 102 should have sufficient resources (e.g., memory) available relative to those needed by the image object 108a, such that the host system is able to grant any legitimate request by the image object 108a.

Consideration should also be given to the instruction set of the virtual machine. For the embodiment of **Figure 1** which is concerned with imaging technology, the instruction set should be optimized for image processing. For instance, there may be no need for instructions that handle character strings,

multi-task execution, or graphical user interfaces. The instruction set should be designed to have enough flexibility to permit the translation of several different native formats for the image data 114a into a common format. Any functions provided in the virtual machine and its instruction set should assume no

5 knowledge of any native formats that can be used for storing the image data 114a.

Yet another consideration for the instruction set is the support of a stack-oriented flow control (call/return/push/pop) with a relatively deep stack. For instance, the host system 102 that implements the virtual machine may provide sufficient resources for an expandable virtual stack that can handle any

10 legitimate stack usage by the object 108a. In addition, the virtual machine should provide programming constraints that can take advantage of parallel processing support if available on the host system 102, particularly because parallel processing may be a key performance enhancement for image processing algorithms. Finally, the virtual machine may also provide flexible exception

15 handling with defined default behavior for every potential exception. Support for high level flow control (if/for/while/case) is also desirable to simplify programming as much as possible and minimize flow control overhead in the object 108a. Flexible type checking is also another feature that may be necessary when casting from one type of variable to another.

20 Exemplary Virtual Machine - The Virtual Image Processor

The virtual machine in one embodiment may be the virtual image processor (VIP) described below. In that embodiment, no traditional I/O support is contemplated for the VIP as no user intervention is expected. The VIP exists solely to process image data 114a as input in a native format and to produce

25 image data as output in the common format. The memory resources required

for such a task would normally be provided by the host system 102, so that the programmer of the image object 108a need not be cognizant of exactly how the memory resources are provided.

As will be apparent from the description below, the instruction set of the
5 VIP may bear little resemblance to that of a conventional hardware
microprocessor. This is in part because of the desire to reduce the code size of
method 115a and because of the emphasis on mostly mathematical processing of
image data 114a. The instruction set of the VIP should allow operations required
for image processing to be specified in the minimum possible space. For
10 instance, flow control instructions can be greatly simplified over those in
conventional microprocessors, because a relatively large amount of associated
complexity may be built into the VIP rather than the instruction set. A single
instruction can thus trigger events that would be too complex for
implementation in the conventional hardware microprocessor. Also,
15 instructions for the VIP may have variable numbers of operands and imply far
more than they would in a hardware microprocessor. Mathematical expressions
need not be broken down into a sequence of data movement instructions with
discrete math operations designed to use a limited set of physical registers.
Instead, expressions are directly represented in the image method 115a and
20 interpreted by the VIP. Assignment expressions may be represented as a target
register followed by the expression in Reverse Polish Notation (RPN). Operators
may therefore be encoded directly into the image method 115a as components of
expressions rather than as machine instructions. The rules governing type cast
and automatic conversions may be implicit in any expression.

There are seven program control instructions and five data manipulation instructions that are contemplated in this embodiment:

Instruction	Description
goto	Transfer program control
ifgoto	Conditionally transfer program control
select	Indexed program control transfer
call	Transfer program control to a sub program
return	Return program control from a sub program
exit	Terminate execution of the VIP program normally
abort	Terminate execution of the VIP program abnormally
push	Push a primitive data variable onto the data stack
pop	Pop a primitive data variable from the data stack
move	Move data and optionally evaluate an expression
create	Create a new data element for program use
destroy	Release a created data element

5 The program control instructions are used to direct the flow of program execution. Unlike in a conventional microprocessor there are not a large number of conditional branch instructions. Also, there is direct support for an indexed jump (corresponding to the **switch** statement in the C programming language). Target address sizes are variable depending on the overall size of the
10 method 115a.

The data movement instructions are the ones that actually do the work of translating and processing an image. In one embodiment, there are no instructions that directly implement math or logic operations because these operations are embedded in the operand list associated with each instruction.
15 Also, unlike a conventional microprocessor, the operand list for an instruction can have a variable number of operands.

The **push** instruction pushes one data variable onto the data stack. Note that the data stack is distinct from the program flow control stack. The **push**

instruction can only be used to push individual primitive data types onto the data stack. That is, it cannot be used to push entire arrays onto the stack. The format of the **push** instruction is an operation code followed by the index of the data variable representing the data to push onto the stack. The type and size of the data variable are encoded into the operand.

The **pop** instruction pops one data variable from the data stack. The associated data variable need not be of the same type as the data previously pushed onto the stack. If the variable type does not match the existing data type at the top of the stack, an automatic type cast will be performed according to the same rules governing type cast for the higher level programming language being used to define the method 115a.

The **move** instruction transfers data from a source to a destination, optionally evaluating an expression encoded in the source. That is, the source need not be a single variable, and can be an arbitrarily complex expression encoded using Reverse Polish Notation (RPN).

The **create** instruction dynamically creates a new data variable and assigns a reference to that variable. The variable created can be a single primitive data type or an array of primitive data variables. The **destroy** instruction releases memory allocated for data variables by the **create** instruction. A compiler for the higher level programming language used to define the method 115a may use the **create** and **destroy** instructions to manage dynamic memory for automatic variables used in sub-programs of the method 115a. This is distinct from the normal practice in the C programming language of using the stack to create

automatic variables. Because the VIP is not intended for implementation as a physical microprocessor there need not be any corollary of a hardware stack.

00443333 00000000

VIP High-Level Language

Because the instruction set for the VIP is optimized for a virtual implementation of an image processor, a low level programming language such as assembly for the VIP may bear little resemblance to that of conventional assembly language for microprocessors. It may also be impractical to implement a full version of a high level language such as C or C++ for use in defining the method 115a, because of the limitations inherent in the imaging-suited optimizations discussed above. Therefore an appropriate compromise may be a subset of the C language. Programs written for the VIP using such a subset of C may have no link step. Rather, they would be compiled as a single file, or as a series of include files (ultimately a single file).

Declarations

The data types allowed are those used with the instruction set discussed earlier. Pointers are to a defined type (i.e. no **void** pointers). A pointer to a pointer would be allowed but not more than double indirection. Multidimensional arrays are declared the same as in conventional C. The **typedef** and **struct** declarators behave as expected. Complex data types such as **struct** can be used as elements in arrays. Some conventional functions, such as bit fields, are essentially meaningless in the context of the VIP embodiment, and therefore for the sake of simplicity bit fields are not allowed.

Type Qualifiers

The **const** type qualifier is allowed in the VIP, but the **volatile** qualifier has no meaning as variables are not visible to any processes outside the scope of a VIP program (e.g., method 115a). Techniques for handling complex data types

such as **struct** are not provided in the method 115a, but are inferred from the structure definitions in the compiled VIP code. This is a key difference between the VIP and a conventional microprocessor. For example:

```
5      struct
      {
        double real_part;
        double imaginary_part;
      } complex[10];
10
      short index=1;

      complex[index].real_part=0.0;
```

With a conventional compiler, the assignment statement shown in the code segment above would be broken down into a series of instructions to calculate the address of the structure field based on the known displacements of its constituent parts from its base address. With the VIP, however, there is no direct corollary of a linear address space as every data item is assigned a unique ID number. The data items may or may not be contiguous in memory. The VIP resolves the complex data type reference into a single ID number that uniquely distinguishes the desired variable. The VIP can distinguish between ID numbers and array indices, so there is no ambiguity. The preceding example is an array of structures, each structure containing two data items. After compilation, this could be represented as:

```
double real_part[10];
double imaginary_part[10];
```

This would result in the equivalent assignment statement:

```
real_part[index]=0.0;
```

In this manner the compiler can break down structure definitions and references into primitive types compatible with the storage scheme used by the VIP.

Storage Class

5 As discussed above, the structure of complex data types is carried in the data rather than the executable code, and data access is resolved by reference rather than by address calculation. There are two exceptions to this in the VIP embodiment: the input data buffer for receiving the image data 114a and the output data buffer for holding the translated image data. These are contiguous
10 arrays of bytes, the size of which are defined at the time a source file is compiled into a method 115a. They are distinguished from other storage types by a storage class not included in standard C.

 There is one input buffer and one output buffer. Any variable declared with the input storage class refers to the input buffer. Any variable declared with
15 the output storage class refers to the output buffer. All types defined with one of these storage classes are mapped to the respective memory array, with respect to its beginning. Two special variables are set by the loader before the method 115a is given control of the VIP: `input_size` and `output_size`. These define the length in bytes of their respective buffers, and can be used without first declaring them
20 in the method 115a (they are valid by definition). In this respect they are similar to the `argc` and `argv` variables defined in standard C, which are not valid for the VIP (they serve no purpose). Using these storage classes and special variables the programmer can map any data type to the input or output buffers. For example:

```
25       input unsigned integer raw_data[input_size/4];  
  
      //Declares an array of integers representing the entire raw image from  
      image data 114a;
```

```
input float raw_fdata[input_size/4];
```

```
5 //Declares an array of floats representing the entire raw image from  
image data 114a.
```

The variables `raw_data[n]` and `raw_fdata[n]` refer to the same data, but are interpreted as different types.

10 A single method 115a can process multiple images of different sizes by having the loader initialize the input and output buffers and size variables for each image. The simplest possible image method would be to copy the input buffer directly to the output buffer because the imaging device 104a produced images in the common format. For example:

```
15 main()  
{  
input byte raw[input_size];  
output byte finished[output_size];  
finished=raw;  
20 }
```

Note the similarity of the assignment statement to the C++ class assignment syntax. This is a deviation from standard C syntax to allow for implicit parallel processing.

Arrays

25 Any array type that is used in an assignment statement or in an expression without an explicit index implies that the operation is to apply to the entire array, and that there are no data dependencies among the elements of the array. Consider the following code fragment:

```
30 short x[256];  
short y[256];  
x=y*1.414;
```

This would multiply by the constant 1.414 every element in the y array and assign the results to the corresponding elements in the x array. If the array sizes do not match the operation would be repeated for the number of elements in the smaller array. Implicit in this is that the operation and assignment can be

5 performed in parallel if possible. There is no particular indexing order implied by this statement. An equivalent notation would be:

`x[]=y[]*1.414;`

This allows multidimensional arrays to be processed in parallel for a single

10 dimension if desired. For example:

`const short column=640;
const short row=480;
unsigned byte image[column][row];`

15 `image[][12]=255;`

This code fragment would assign the value 255 to all elements in row 12 of the image array. When compiling this type of high level code for a conventional microprocessor, the compiler might generate a sequence of machine instructions to explicitly carry out the loop operation implicit in the statement. The VIP

20 compiler, however, simply records an implicit loop operation and any relevant parameters required, and the VIP itself infers the necessary index repetitions.

This reduces the size of stored code in the method 115a at the expense of performance. Such tradeoff may be acceptable if the performance of the host

25 system 102 on which the VIP is installed is sufficient to execute the required functions at an acceptable rate.

To summarize, various embodiments of the invention have been described as techniques for improving the portability of still image data using the concept of image objects. It is, however, expected that the described image object

and abstract machine technique may be applied to other types of data which are also coherent and bounded, such as certain types of audio or video. In those instances, the image method may be replaced with an audio/video codec typically used for compression and decompression of the audio/video sequence.

5 The embodiments of the invention described above are, of course, subject to other variations in structure and implementation. For instance, although **Figure 1** illustrates a single object in each imaging or storage device, this is done only to ease the explanation of how the invention works in the given embodiment. Indeed, one of ordinary skill in the art will recognize that multiple
10 objects may be created by each device and transferred to the host system as needed to represent multiple images generated by each device. Also, many other alternatives to the computing architectures of **Figure 3** and **4** are possible which achieve image capture and image object formation. Therefore, the scope of the invention should be determined not by the embodiments illustrated but by the
15 appended claims and their legal equivalents.

CLAIMS:

What is claimed is:

1 1. An article comprising:

2 a machine-readable medium having instructions that when executed by a
3 processor cause the step of
4 associating first image data and first method as part of an image
5 object, the first method for being executed by an abstract machine to obtain first
6 translated image data based upon the first image.

1 2. The article of claim 1 wherein the machine readable medium
2 further comprises instructions that when executed by the processor cause the
3 further step of:
4 associating second image data with the first method as part of the
5 object, the first method for being executed by the abstract machine to obtain
6 second translated image data based upon the second image data.

1 3. The article of claim 1 wherein the machine readable medium
2 further comprises instructions that when executed by the processor cause the
3 further step of:
4 associating second image data and second method as part of a
5 second object, the second method for being executed by the abstract machine to
6 obtain second translated image data based upon the second image data.

1 4. The article of claim 1 wherein the first translated data is in the same
2 format as the first data.

1 5. An article comprising
2 a machine-readable medium having instructions that when
3 executed by a processor cause the steps of
4 configuring a data processing system to receive first and
5 second objects from first and second imaging devices, respectively, the objects
6 having first and second image data and corresponding methods; and
7 an abstract machine executing the corresponding methods of
8 each object to obtain first and second translated image data based upon the first
9 and second image data, respectively.

1 6. The article of claim 5 wherein the first and second translated image
2 data are in the same image file format.

1 7. A method comprising:
2 transferring an image object having first image data associated with
3 a first method to a processing system; and
4 an abstract machine in said processing system executing the first
5 method for generating first translated image data based upon the first image data.

1 8. The method of claim 7 further comprising:
2 transferring a second object having second image data associated
3 with a second method to the processing system, the first and second image data
4 being in different formats; and
5 the abstract machine executing the second method generating
6 second translated image data based upon the second image data, the first and
7 second translated image data being in the same format.

1 9. The method of claim 7 further comprising:
2 transferring second image data associated with the first method to
3 the processing system; and
4 the abstract machine executing the first method generating second
5 translated image data based upon the second image data, the first and second
6 translated image data being in the same format.

1 10. An imaging device comprising:
2 image sensor for generating sensor data; and
3 memory for storing an image object having first image data being
4 related to the sensor data and first image method for being executed by an abstract
5 machine to obtain translated first image data based upon the first image data.

1 11. The imaging device of claim 10 wherein the first image data is the
2 sensor data.

1 12. The imaging device of claim 10 further comprising
2 a processor; and
3 second memory having instructions that when executed by the
4 processor cause processing the sensor data into the first image data.

1 13. The imaging device of claim 12 wherein the processing comprises
2 performing an image processing methodology on the sensor data.

1 14. The imaging device of claim 10 further comprising:
2 logic circuitry for processing the sensor data into the first image
3 data.

1 15. The imaging device of claim 14 wherein the logic circuitry performs
2 a color interpolation algorithm on the sensor data.

1 16. The imaging device of claim 10 further comprising:
2 interface to a communication medium for transferring the first
3 image data and the first method to a processing system separate from the

4 imaging device, the processing system being configured with said abstract
5 machine.

1 17. The imaging device of claim 10 wherein the image object comprises
2 a TIFF file, the TIFF file comprising the first image data and the first image
3 method.

1 18. The imaging device of claim 10 wherein the translated first image
2 data is part of an image file being in the Device Independent Bitmap (DIB)
3 format.

1 19. The imaging device of claim 10 wherein the first image data and the
2 translated first image data have the same image file format.

1 20. A data processing system comprising:

2 a processor;

3 memory coupled to the processor and having instructions that
4 when executed by the processor cause the steps of

5 configuring the system to receive first and second objects
6 from first and second imaging devices, respectively, each object having image
7 data and a corresponding method; and

8 an abstract machine executing the corresponding method of
9 each object to obtain corresponding translated data based upon the image data.

ABSTRACT

Techniques for storing and translating digital images of different native formats in a way that makes it unnecessary for independent manufactures of imaging devices such as digital cameras to agree on factors related to production/capture and processing of the images. Image data in a native format and an associated method are combined as part of an image object, within the meaning of classical object-oriented technology, and transferred to the host system. An abstract machine, such as a virtual machine, in the host system acts as a virtual image processor and executes the method to obtain image data in a common format. The method in each image object operates on its corresponding image data to yield an image in a common format.

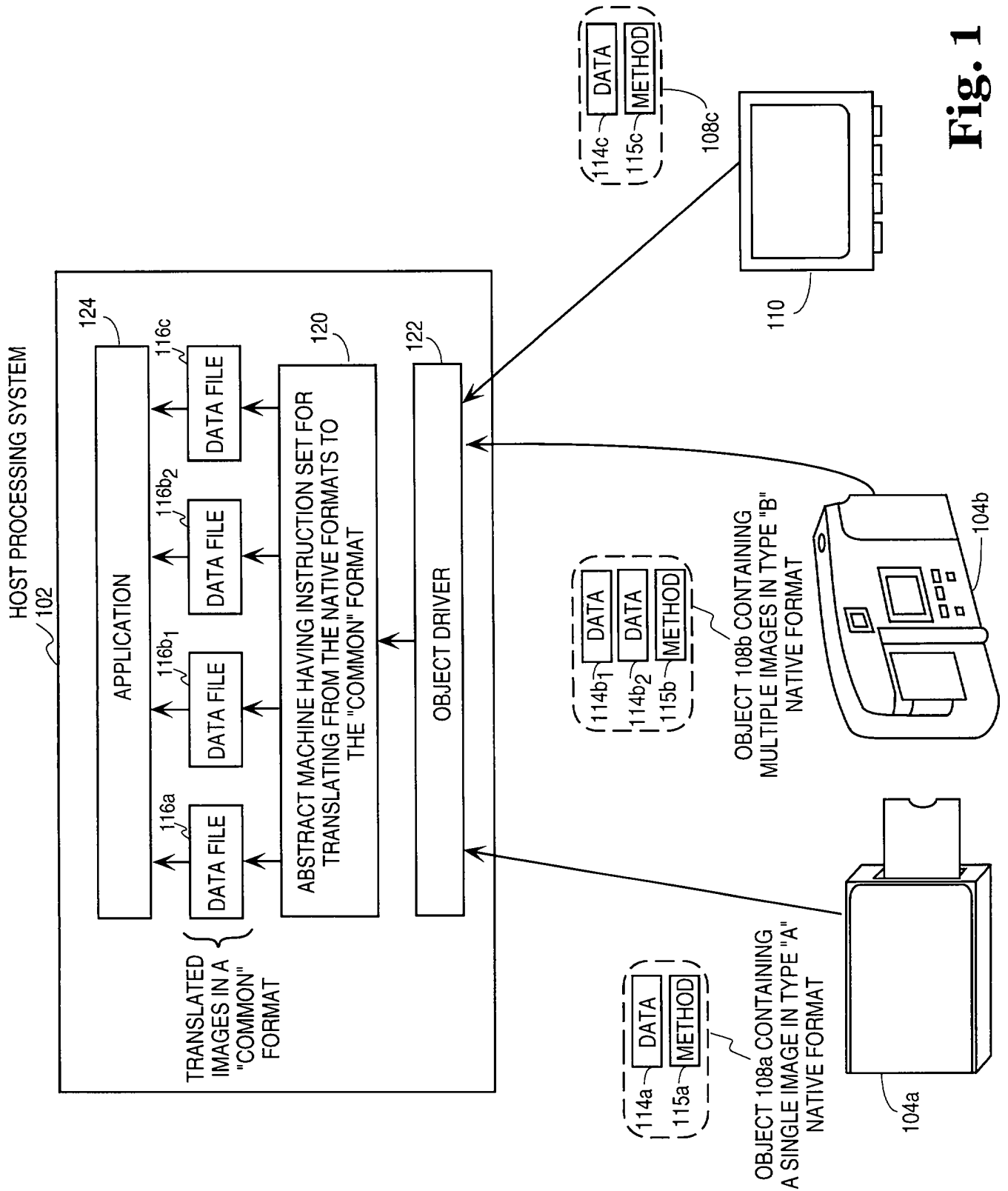


Fig. 1

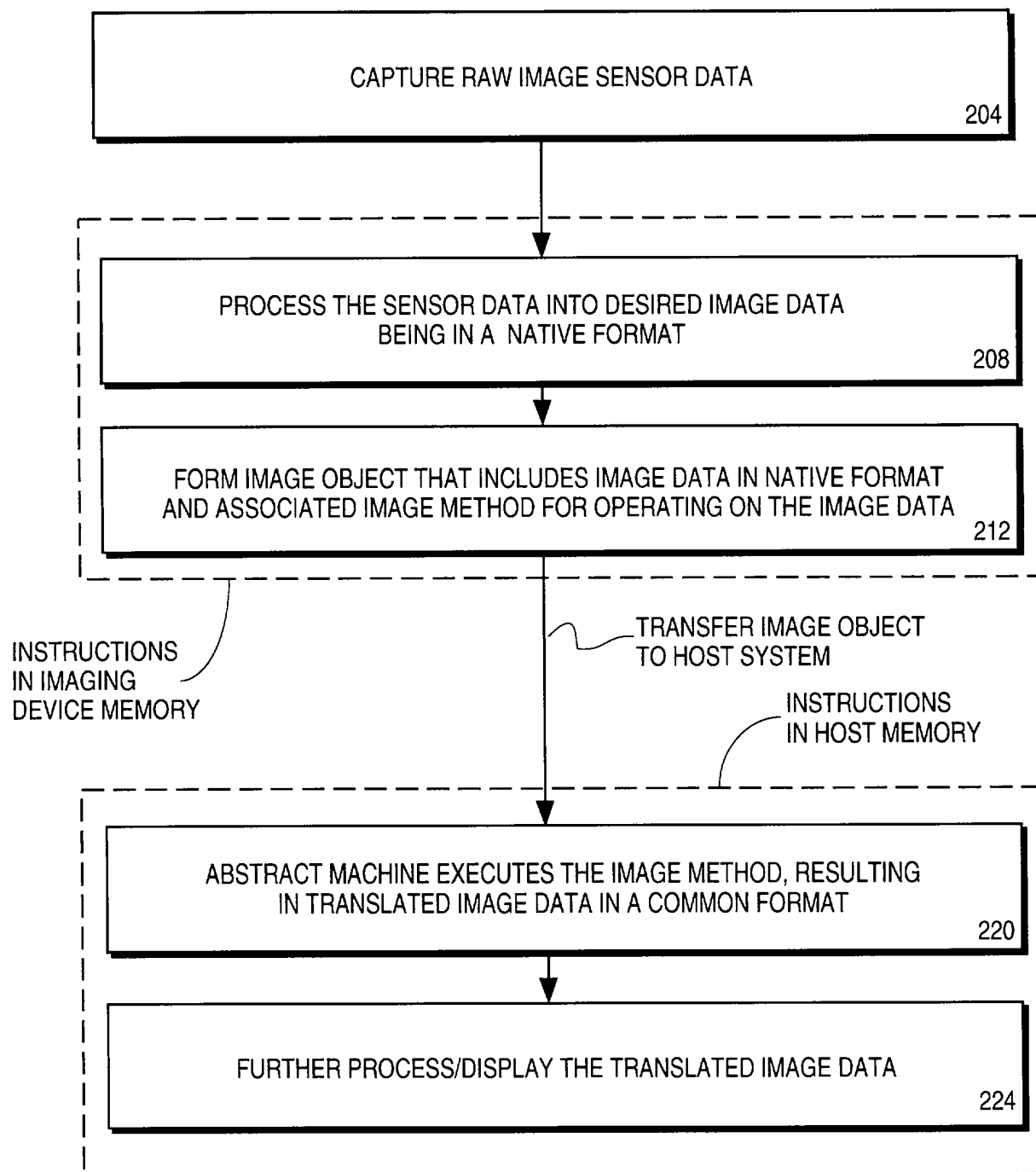
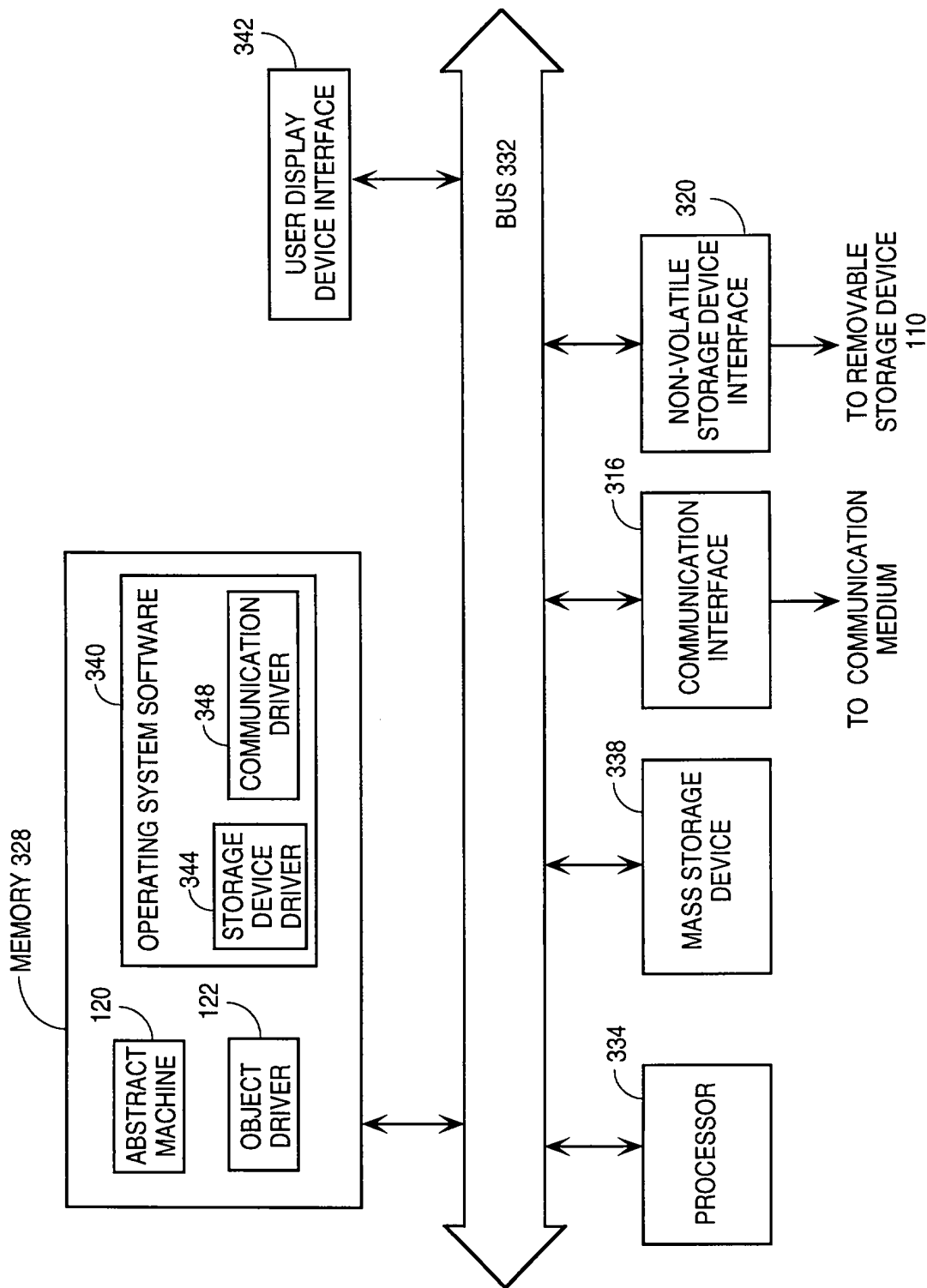
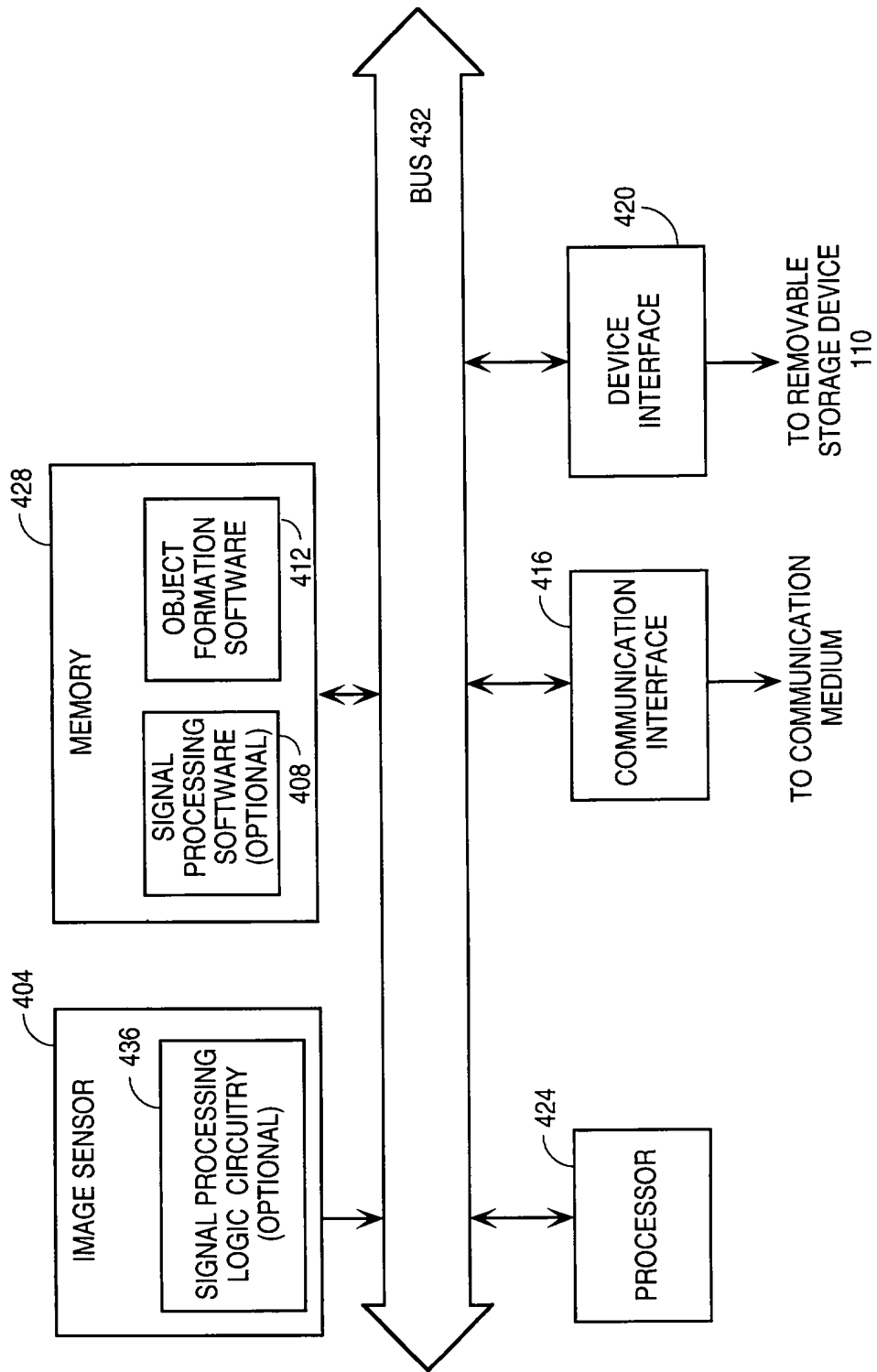


Fig. 2



HOST PROCESSING SYSTEM 102

Fig. 3



IMAGING DEVICE 104a

Fig. 4

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or any original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

IMPROVING THE PORTABILITY OF DIGITAL IMAGES

the specification of which ☒ is attached hereto.
☐ was filed on _____ as
United States Application Number _____
or PCT International Application Number _____
and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):

APPLICATION NUMBER	COUNTRY (OR INDICATE IF PCT)	DATE OF FILING (day, month, year)	PRIORITY CLAIMED
			<input type="checkbox"/> No <input type="checkbox"/> Yes
			<input type="checkbox"/> No <input type="checkbox"/> Yes
			<input type="checkbox"/> No <input type="checkbox"/> Yes

I hereby claim the benefit under Title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below:

APPLICATION NUMBER	FILING DATE

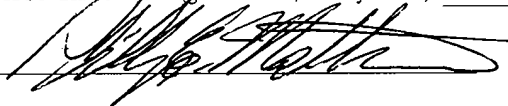
I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION NUMBER	FILING DATE	STATUS (ISSUED, PENDING, ABANDONED)

I hereby appoint BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, a firm including: Farzad E. Amini, Reg. No. P42,261; Aloysius T. C. AuYeung, Reg. No. 35,432; William Thomas Babbitt, Reg. No. 39,591; Carol F. Barry, P41,600; Jordan Michael Becker, Reg. No. 39,602; Bradley J. Bereznak, Reg. No. 33,474; Michael A. Bernadicou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; Gregory D. Caldwell, Reg. No. 39,926; Kent M. Chen, Reg. No. 39,630; Lawrence M. Cho, Reg. No. 39,942; Yong S. Choi, Reg. No. P43,324; Thomas M. Coester, Reg. No. 39,637; Roland B. Cortes, Reg. No. 39,152; Barbara Bokanov Courtney, Reg. No. P42,442; William Donald Davis, Reg. No. 38,428; Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Tarek N. Fahmi, Reg. No. P41,402; James Y. Go, Reg. No. 40,621; Richard Leon Gregory, Jr., P42,607; Dinu Gruia, Reg. No. P42,996; David R. Halvorson, Reg. No. 33,395; Thomas A. Hassing, Reg. No. 36,159; Phuong-Quan Hoang, P41,839; Willmore F. Holbrow III, Reg. No. P41,845; George W. Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; Dag H. Johansen, Reg. No. 36,172; William W. Kidd, Reg. No. 31,772; Tim L. Kitchen, Reg. No. P41,900; Michael J. Mallie, Reg. No. 36,591; Paul A. Mendonsa P42,879; Darren J. Milliken, P42,004; Thinh V. Nguyen, Reg. No. P42,034; Kimberley G. Nobles, Reg. No. 38,255; Michael A. Proksch P43,021; Babak Redjaian, Reg. No. P42,096; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Anand Sethuraman, Reg. No. P43,351; Charles E. Shemwell, Reg. No. 40,171; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Allan T. Sponseller, Reg. No. 38,318; Steven R. Sponseller, Reg. No. 39,384; Geoffrey T. Staniford, P43,151; Judith A. Szepesi, Reg. No. 39,393; Vincent P. Tassinari, Reg. No. P42,179; Edwin H. Taylor, Reg. No. 25,129; George G. C. Tseng, Reg. No. 41,355; Lester J. Vincent, Reg. No. 31,460; John Patrick Ward, Reg. No. 40,216; Stephen Warhola, P43,237; Ben J. Yorks, Reg. No. 33,609; and Norman Zafman, Reg. No. 26,250; my attorneys; and Amy M. Armstrong, Reg. No. P42,265; Robert Andrew Diehl, Reg. No. P40,992; and Edwin A. Sloane, Reg. No. 34,728; my patent agents, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (310) 207-3800, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor (given name, family name) Phillip E. Mattison

Inventor's Signature  Date 8/14/98

Residence Gilbert, Arizona Citizenship USA
(City, State) (Country)

P. O. Address 734 E. Comstock

Gilbert, Arizona 85296